Rekursif

- Rekursif adalah salah satu metode dalam dunia matematika dimana definisi sebuah fungsi mengandung fungsi itu sendiri.

  Dalam dunia pemrograman, rekursi diimplementasikan dalam sebuah fungsi yang memanggil dirinya sendiri

- Contoh fungsi rekursif misalnya adalah fungsi pangkat, faktorial, dan barisan fibonacci.
- Dalam fungsi pangkat xy , kita tahu bahwa semua bilangan selain 0, jika dipangkatkan dengan 0 nilainya sama dengan 1. Jika x dipangkatkan dengan y, dengan y lebih dari 0, maka hasilnya sama dengan x dikalikan dengan x dipangkatkan y – 1.

Jika dituliskan dalam notasi matematika definisinya

$$x^y = 1, jika\ y = 0$$

$$x^y = x * x^{y-1}, jika\ y > 0$$

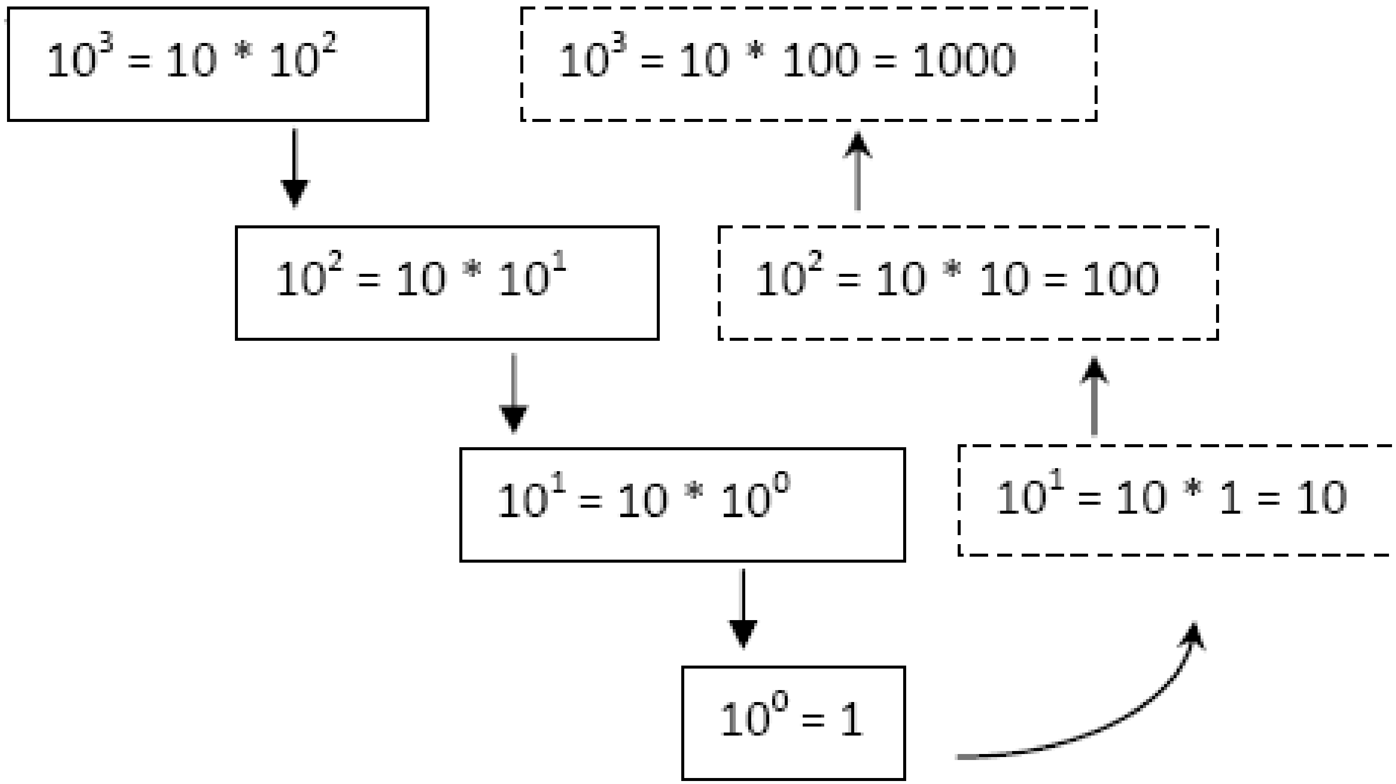
Kita lihat di atas pada definisi y > 0, bentuk pemangkatan muncul kembali di sisi kanan. Itulah yang disebut <u>rekursif.</u>

Definisi rekursif selalu dimulai dengan kasus penyetop, penghenti, atau kasus dasar dari suatu permasalahan, dalam hal ini terjadi ketika nilai $y = 0$.

Definisi rekursif yang lebih kompleks mengandung inti dari permasalahan yang akan dipecahkan, namun lebih sederhana. Dalam hal ini yang tadinya x dipangkatkan dengan y, kini bentuk pemangkatan menjadi lebih sederhana, yaitu $y – 1$.

Hal ini dimaksudkan untuk “menggiring” masalah kompleks ke kasus dasar atau penyetop rekursinya.

Untuk x = 10 dan y = 0, hasil dari x^y adalah 1. Untuk x = 10 dan y = 3 hasilnya dapat digambarkan sebagai
berikut:



Ide dasar dalam memecahkan suatu masalah dengan rekursif adalah sebagai berikut:

Contoh Lain

Mari kita lihat contoh rekursif yang jauh lebih sederhana,Masalah yang akan dipecahkan adalah memotong roti tawar tipis-tipis sampai habis. Jika masalah ini akan dipecahkan secara rekursif, maka solusinya adalah:

1. Jika roti sudah habis atau potongannya sudah paling tipis, pemotongan roti selesai
2. Jika roti masih bisa dipotong, potong tipis dari tepi roti tersebut, lalu lakukan prosedur 1 dan 2 untuk sisa potongannya.

Contoh Program

```
program faktorial;
uses wincrt;
var
   faktor :real;
   i,n :integer;
begin
   write('Masukkan bilangan n =');readln(n);
   faktor:=1;
   for i:= 2 to n do {Menghitung n faktorial}
   faktor:=faktor*i;
   writeln(n,' Faktorial = ',faktor:0:0);
end.
```

Hasil Program

Masukan Bilangan n = 3
3 Faktorial 6

Masukan Bilangan n = 2
2 Faktorial 1

Kita dapat menuliskan fungsi penghitung factorial seperti dibawah ini

```
1.    int Faktorial(int n)
2.  {
3.      if ((n == 0) || (n == 1 ))
4.           return (1);
5.      else
6.           return (n * Faktorial(n-1));
7.  }
```

- Pada *baris 3* dari fungsi diatas,
  nilai n dicek sama dengan 0 atau 1,
  jika ya, maka fungsi mengembalikan nilai 1 {*baris 4*},
  jika tidak, fungsi mengembalikan nilai n * Faktorial (n -1) {*baris 6*}
- disinilah letak proses rekursif itu, perhatikan fungsi factorial ini memanggil dirinya sendiri tetapi dengan parameter (n-1)

Barisan yang didefinisikan secara rekursif

Contoh:

Barisan bilangan pangkat dari 2

$$a_n = 2^n \text{ untuk } n = 0, 1, 2, \ldots \,.$$

Barisan ini dapat didefinisikan secara rekursif:

$$a_0 = 1$$

$$a_{n+1} = 2a_n \quad \text{untuk } n = 0, 1, 2, \ldots$$

Langkah-langkah untuk mendefinisikan barisan secara rekursif:

1. Langkah basis: Spesifikasi anggota awal.
2. Langkah rekursif: Berikan aturan untuk membangun anggota baru dari anggota yang telah ada.

Contoh barisan yang didefinisikan secara rekursif

Berikan definisi rekursif dari $a_n = r^n$, dengan $r \in \mathbb{N}$, $r \neq 0$ dan n bilangan bulat positif.

Solusi:

Definisikan $a_0 = r^0 = 1$

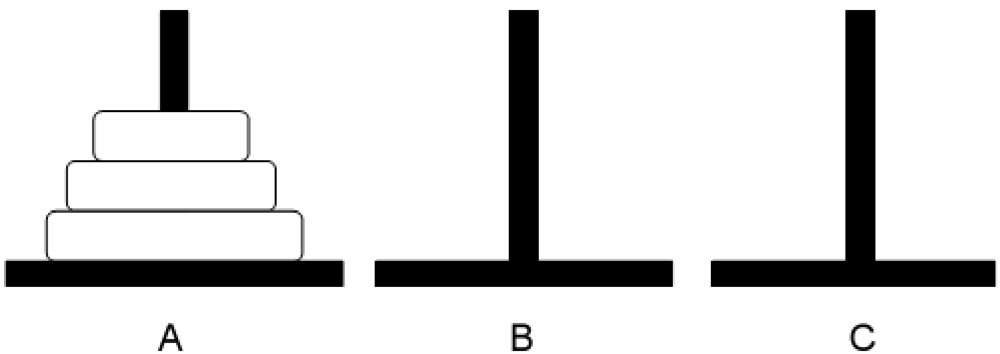
dan $a_{n+1} = r \cdot a_n$ untuk n = 0, 1, 2, ...

- **Contoh:** definisi rekursif himpunan Ekspresi Aritmatika **EA**
- <u>Basis:</u> **1, 2, 3, 4, 5** $\in$ **EA**
- <u>Rekursif:</u> jika **a** $\in$ **EA** dan **b** $\in$ **EA**, maka
- **a + b** $\in$ **EA**
- **a – b** $\in$ **EA**
- **a** $\times$ **b** $\in$ **EA**
- **a** $\div$ **b** $\in$ **EA**

Legenda Menara Hanoi
(oleh Edouard Lucas abad 19)

- Seorang biarawan memiliki 3 menara.
- Diharuskan memindahkan 64 piringan emas.
- Diameter piringan tersebut tersusun dari ukuran kecil ke besar.
- Biarawan berusaha memindahkan semua piringan dari menara pertama ke menara ketiga tetapi harus melalui menara kedua sebagai menara tampungan.
- Kondisi:
  - Piringan tersebut hanya bisa dipindahkan satu-satu.
  - Piringan yang besar tidak bisa diletakkan di atas piringan yang lebih kecil.
- Ternyata : mungkin akan memakan waktu sangat lama (sampai dunia kiamat).
- Secara teori, diperlukan 2^{64}-1 perpindahan. Jika kita salah memindahkan, maka jumlah perpindahan akan lebih banyak lagi.
- Jika satu perpindahan butuh 1 detik, maka total waktu yang dibutuhkan lebih dari 500 juta tahun !!.

Menara Hanoi



Jika piringan ada 3, berapa kali jumlah perpindahan?

Menara Hanoi

Untuk memindahkan n piringan dari tiang 1 ke tiang 3:

1. **Pindahkan (n-1) piringan dari tiang 1 ke tiang 2**
2. **Pindahkan 1 piringan (terbesar) dari tiang 1 ke tian 3**
3. **Pindahkan (n-1) piringan dari tiang 2 ke tiang 3**

H(n) : untuk memindahkan n piringan

$$\left.\begin{array}{ll} 1. & H(n-1)\ \text{pemindahan} \\ 2. & 1\ \text{pemindahan} \\ 3. & H(n-1)\ \text{pemindahan} \end{array}\right\}\ \text{total ada}\ \ 2H(n-1) + 1$$

Menara Hanoi

- Rumus : **H(n) = 2 H(n – 1) + 1**
- Algoritma:
  - Jika n==1, pindahkan pringan dari A ke C
  - Jika tidak:
    - Pindahkan n-1 piringan dari A ke B menggunakan C sebagai tampungan
    - Pindahkan n-1 piringan dari B ke C menggunakan A sebagai tampungan

Fungsi rekursif:

Contoh: fungsi Fibonacci

Basis: $fib(0) = 0; \; fib(1) = 1$

Rekursif: $fib(n) = fib(n - 1) + fib(n - 2)$

Ditulis dengan cara lain:

$$fib(n) = \begin{cases} n & \text{jika } n = 0, 1 \\ fib\,(n-1) + fib\,(n-2) & \text{jika } n > 1 \end{cases}$$

Fungsi yang didefinisikan secara rekursif

Langkah-langkah untuk mendefinisikan fungsi dengan domain bilangan cacah:

1. Langkah basis: Definisikan nilai fungsi pada saat nol.
2. Langkah rekursif: Berikan aturan untuk mencari nilai fungsi untuk setiap bilangan bulat berdasarkan nilai fungsi pada bilangan bulat yang lebih kecil.

Definisi seperti itu disebut rekursif atau definisi induktif.

Contoh fungsi yang didefinisikan secara rekursif

$f(0) = 3$

$f(n + 1) = 2f(n) + 3$

Maka

$$f(0) = 3$$

$$f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$$

$$f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$$

$$f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$$

$$f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$$

Contoh fungsi yang didefinisikan secara rekursif (2)

Bagaimana kita dapat mendefinisikan fungsi faktorial $f(n) = n!$ secara rekursif?

$$f(0) = 1$$

Karena $(n+1)! = n!\,(n+1)$ maka

$$f(n + 1) = (n + 1)f(n)$$

$f(0) = 1$

$f(1) = 1\,f(0) = 1 \cdot 1 = 1$

$f(2) = 2\,f(1) = 2 \cdot 1 = 2$

$f(3) = 3\,f(2) = 3 \cdot 2 = 6$

$f(4) = 4\,f(3) = 4 \cdot 6 = 24$

Fibonacci

- Deret Fibonacci adalah suatu deret matematika yang berasal dari penjumlahan dua bilangan sebelumnya.

  1, 1, 2, 3, 5, 8, 13, 21, …
- Rumus Fibonacci :

$$\mathbf{f(n) = f(n - 1) + f(n - 2)}$$

$$f(6) = f(6\text{-}1) + f(6\text{-}2)$$

$$8 \quad = \quad 5 \quad + \quad 3$$

Bilangan Fibonacci

- Fungsi lain yang dapat diubah ke bentuk rekursif adalah perhitungan Fibonacci. Bilangan Fibonacci dapat didefinisikan sebagai berikut:

  $f_n = f_{n-1} + f_{n-2}$ untuk $n > 2$

  $f_1 = 1$

  $f_2 = 1$

  Berikut ini adalah barisan bilangan Fibonacci mulai dari n=1

  1 1 2 3 5 8 13 21 34

Algoritma Fibonacci yang dipakai

```
Function Fibonacci(input n:integer) → integer
Deklarasi Lokal
  {tidak ada}
Deskripsi
  If (n ==1 || n==2) Then
     return (1)
  Else
     return (Fibonacci(n-1)+Fibonacci(n-2))
  Endif
```

Contoh terkenal: Bilangan Fibonacci

$f_0 = 0, f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$, $n=2,3,4,\ldots$

$$
\begin{aligned}
f_0 &= 0 \\
f_1 &= 1 \\
f_2 &= f_1 + f_0 = 1 + 0 = 1 \\
f_3 &= f_2 + f_1 = 1 + 1 = 2 \\
f_4 &= f_3 + f_2 = 2 + 1 = 3 \\
f_5 &= f_4 + f_3 = 3 + 2 = 5 \\
f_6 &= f_5 + f_4 = 5 + 3 = 8
\end{aligned}
$$

Tunjukkan bahwa untuk $n \geq 3$,

$f_n < \alpha^n$ dengan $\alpha = (1+\sqrt{5})/2$.

Himpunan yang didefinisikan secara rekursif

Langkah-langkah dalam mendefinisikan suatu himpunan secara rekursif:

1. Langkah basis:

   Spesifikasi koleksi awal dari anggota

2. Langkah rekursif:

   Mendefinisikan aturan konstruksi anggota baru dari anggota yang telah diketahui

Contoh himpunan yang didefinisikan secara rekursif

Misalkan S didefinisikan secara rekursif oleh:

$3 \in S$

$(x+y) \in S$ jika $x \in S$ dan $y \in S$

Maka S adalah himpunan bilangan bulat positif yang habis dibagi 3.

Bukti:

Misalkan A himpunan yang beranggotakan semua bilangan bulat positif yang habis dibagi 3.

Untuk membuktikan bahwa A = S, harus ditunjukkan

$A \subseteq S$ and $S \subseteq A$.

Bagian I: Akan dibuktikan $A \subseteq S$, yaitu menunjukkan bahwa setiap bilangan bulat positif yang habis dibagi 3 ada di S (dengan menggunakan induksi matematika).

Contoh himpunan yang didefinisikan secara rekursif (2)

Misalkan P(n): proposisi “3n anggota S”.

1. Langkah basis: P(1) benar, karena $3 \in S$.
2. Langkah induktif:

   Asumsikan P(k) benar, yaitu $3k \in S$.

   Akan ditunjukkan P(k+1) juga benar, yaitu

   $3(k+1) \in S$

   Karena $3k \in S$ dan $3 \in S$, berdasarkan definisi rekursif dari S, $3k+3 = 3(k+1)$ juga ada di S.
3. Konklusi:

   Jadi, setiap bilangan bulat positif yang habis dibagi 3 ada di S.

Kesimpulan dari bagian I adalah $A \subseteq S$.

Contoh himpunan yang didefinisikan secara rekursif (3)

Akan ditunjukkan $S \subseteq A$ dengan menggunakan definisi rekursif dari S.

Langkah basis:

Akan ditunjukkan setiap anggota awal S ada di A.

Karena 3 habis dibagi 3 maka $3 \in A$.

Langkah rekursif:

Akan ditunjukkan bahwa setiap bilangan bulat yang dibangun dengan mengunakan langkah rekursif juga merupakan anggota A, yaitu

$$(x+y) \in A \text{ jika } x,y \in S \text{ (yang diasumsikan } \in A).$$

Jika x dan y keduanya di A, maka $3 \mid x$ dan $3 \mid y$. Akibatnya, $3 \mid (x + y)$.

Kesimpulan dari bagian II adalah $S \subseteq A$.

Jadi, secara keseluruhan, berlaku $A = S$.

Kelebihan dan kelemahan rekursi :

- Kelebihan
  - solusi sangatlah efisien
  - dapat memecahkan masalah yang sulit dengan tahapan yang mudah dan singkat

- Kelemahan
  - sulit dipahami
  - perlu stack besar (stack overrun)